

Web Interface HTML Design

This section covers the following topics:

- Given Layout as HTML Table
 - Restrictions
 - Chosen Layout
 - Input Layout
 - Functional Parts
 - How to Use Frames
 - How to Use JavaScript
-

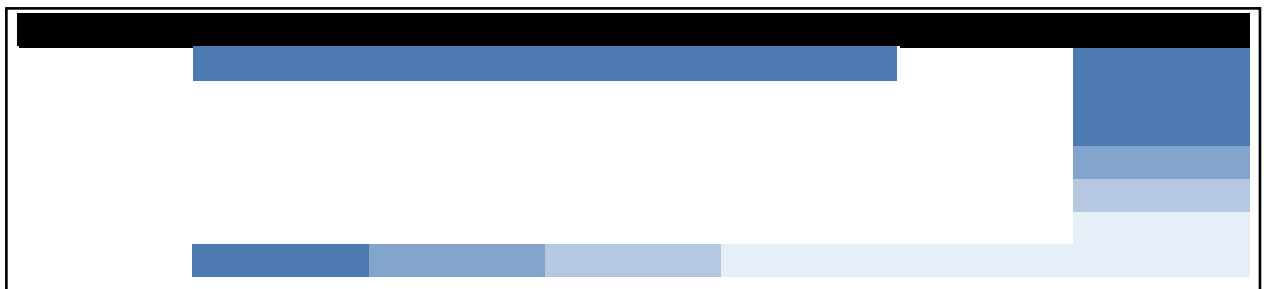
The Given Layout as HTML Table

If an HTML page is to be generated with the given layout, HTML 4.0 with extensive use of Cascading Style Sheets (CSS) is needed. In addition, the user needs a display with more the 256 colors.

If this is available, the figure below shows the result of the design. The page consists of a Table with 11 rows and 7 columns. The border on the left side is realized with one merged table field with a background picture. The picture is repeated in the x direction. All other borders are separate fields with a fixed background color. The area in the middle of the page is a merged area of 5 rows and 5 columns, filled with a background picture. The small space to the right of the headline border is filled with a small picture to give the right appearance.

The pictures used are saved in jpg format, to ensure that the correct color settings are used. For example the picture used for the middle pages used 1036 different colors.

To force the HTML browser to leave the right amount of space, a "single-pixel-GIF" is used. A transparent picture containing only one pixel is resized to the space you need.



HTML Source of the Table

```

<TABLE BORDER="0" CELLSPACING="0" CELLPADDING="0" WIDTH="500">
<!-- 1st line -->
<TR bgcolor='#000000'>
  <!-- black title line >
  <TD COLSPAN=7>&nbsp;</TD>
</TR>
<!-- 2nd line>
<TR VALIGN='MIDDLE'>
  <!-- black field left of the title field>
  <TD bgcolor='#000000'>
    <IMG HEIGHT="5" WIDTH="5" SRC="dot_clear.gif">
  </TD>
  <!-- blue title field >
  <TD BGCOLOR="#4D7AB1" ROWSPAN="4" COLSPAN="4" ALIGN="CENTER"
VALIGN="Middle">
    &nbsp;</TD>
  <!-- black field right of the title field >
  <TD ROWSPAN=2 COLSPAN=2 bgcolor='#000000'>
    <IMG HEIGHT="1" WIDTH="5" SRC="dot_clear.gif">
  </TD>
</TR>
<!-- 3rd line -->
<TR>
<!-- left shaded border >
  <TD ROWSPAN=9 style="background-image:url(bg_fl.jpg);
background-repeat:repeat-x">
    &nbsp;</TD>
</TR>
<!-- 4th line -->
<TR>
  <!-- small field right of the border>
  <TD ROWSPAN=2 style="background-image:url(bg_ro.jpg);
background-repeat:no-repeat">
    &nbsp;</TD>
  <!-- right dark blue border>
  <TD ROWSPAN=2 bgcolor='#4D7AB1'>&nbsp;</TD>
</TR>
<!-- 5th line -->
<TR></TR>
<!-- 6th line -->
<TR>
  <!-- center field of the table >
  <TD WIDTH="480" HEIGHT="600" ROWSPAN=5 COLSPAN=5
style="background-image:url(bg_m.jpg); background-repeat:no-repeat">
    &nbsp;</TD>
  <!-- right border line>
  <TD bgcolor='#4D7AB1'>&nbsp;</TD>
</TR>
<!-- 7st line -->
<TR>

```

```

    <!-- right border line -->
    <TD bgcolor='#4D7AB1'>&nbsp;</TD>
</TR>
<!-- 8st line -->
<TR>
    <!-- right border line -->
    <TD bgcolor='#83A4CD'>&nbsp;</TD>
</TR>
<!-- 9st line -->
<TR>
    <!-- right border line -->
    <TD bgcolor='#B6C7E1'>&nbsp;</TD>
</TR>
<!-- 10st line -->
<TR>
    <!-- right border line -->
    <TD bgcolor='#E6EFF8'>&nbsp;</TD>
</TR>
<!-- 11st line -->
<TR>
    <!-- bottom border line -->
    <TD bgcolor='#4D7AB1' WIDTH=25%>&nbsp;</TD>
    <TD bgcolor='#83A4CD' WIDTH=25%>&nbsp;</TD>
    <TD bgcolor='#B6C7E1' WIDTH=25%>&nbsp;</TD>
    <TD bgcolor='#E6EFF8' WIDTH=25%>&nbsp;</TD>
    <TD bgcolor='#E6EFF8'>&nbsp;</TD>
    <TD bgcolor='#E6EFF8'>&nbsp;</TD>
</TR>
</TABLE>

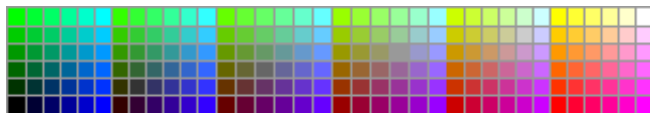
```

Restrictions

For a real web application, nobody would choose the layout above. There are some restrictions that should be considered if not only an Intranet with fixed defined HTTP browser and settings are used. The real world looks different.

Colors

Colors should be used from the so called "Netscape Color Cube" (see the figure below).



These are the 216 colors that the Netscape browser uses in its palette. By using these colors on your web pages, you can be assured that the viewer will see your images exactly as you intended. If you do not use the color cube, Netscape will use it for you.

Even if Internet Explorer with 256 colors is used, these colors work well. Other colors will lead to the same effects. This happens not only to fix coded color setting inside your HTML, the same effect can be seen with pictures.

HTML 4.0 and Cascading Style Sheets

Cascading Style Sheets (CSS) and HTML 4.0 are the current standard of HTML. But only HTML 3.2 is really browser independent. With Internet Explorer 4 and Netscape 4, CSS is implemented, but there are differences.

Fixed Table Size

Our page uses fixed table cell sizes to display the background pictures in the right way. However, every browser allows you to set up the size of the font used.

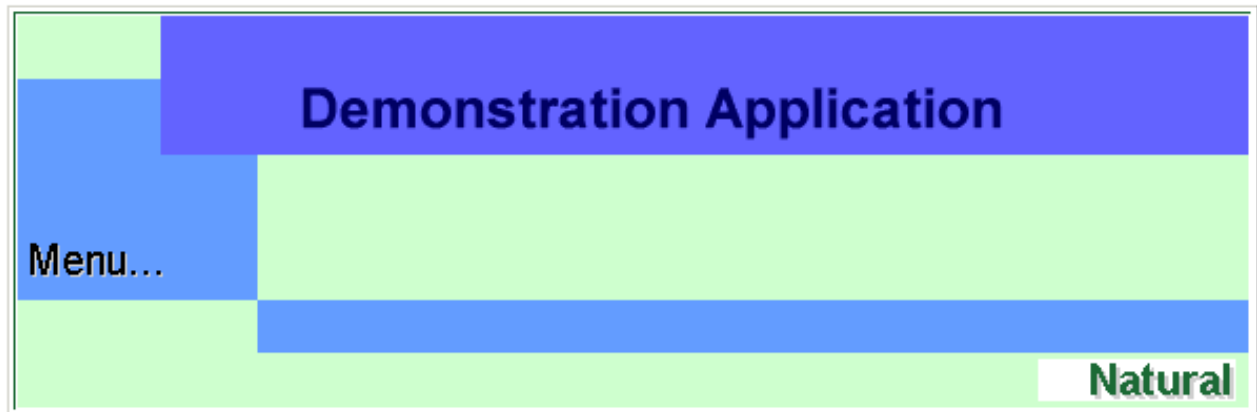
It is very common that pages only look good if a specific font size is used. Other fonts lead to strange effects beginning with text that is displayed at the wrong column and leads to strange looking background pictures.

A table will always resize to display the text inside, regardless of whether a size is given. If more space is needed, the space is taken.

Therefore it is better to set up table cell only with percent settings.

Chosen Layout

The new layout, realized with a table, looks like the figure below.



Input Layout

It is necessary to return input data from the HTML page. Forms enable you to send data back to the HTTP server. Input elements are INPUT, TEXTAREA and SELECT (HTML 3.2). A form should have at least one submit button. The main input screen to select a range of employees may look like this:

from: <input type="text" value="A"/>	to: <input type="text" value="Az"/>	<input type="button" value="Browse Employee"/>
--------------------------------------	-------------------------------------	--

The HTML looks like this:

```
<FORM ACTION="/cgi-bin/nwwcgi.exe/sysweb/nat-env">
from:
<INPUT TYPE="text" NAME="FROM" VALUE="A" SIZE="5">
to:
<INPUT TYPE="text" NAME="TO" VALUE="Az" SIZE="5">
<INPUT TYPE="submit" VALUE="Browse Employee">
</FORM>
```

The code shows that it is only possible to specify one URL to be called. However, the specification is supposed to call different selections. One way to do this is to add more than one form:

from: <input type="text" value="A"/>	to: <input type="text" value="Az"/>	<input type="button" value="Select Employee"/>
from: <input type="text" value="A"/>	to: <input type="text" value="Az"/>	<input type="button" value="Browse Employee"/>
from: <input type="text" value="A"/>	to: <input type="text" value="Az"/>	<input type="button" value="Listing Employee"/>

Another possibility is to call only one program, some kind of dispatcher. This program calls the subprogram needed. The dispatching can be done by the value of the submit button. Therefore, the submit button must have a name, e.g. "TODO". The form then looks like the figure below:

from: to:

If we now combine the given layout with our form, our main page will look like the figure below:

Maintenance of File
MENU

File:

Selection by Name

From: To:

Natural

Using the submit buttons does not lead to a harmonious display, because the appearance of the submit buttons cannot be changed.

However, submit buttons can be replaced by pictures, as in the figure below:

Select... Browse... Listing... from: to:

However, the pictures do not look like submit buttons, or links to another object. Will they do something?

It would be better if a menu could be created with changing images. This cannot be done by using only HTML. If JavaScript can be used, it is possible to change the appearance of a picture if the mouse is over it:

```
script language="JavaScript" type="text/javascript" !-- function imgAct(imgName) { document[imgName].src =
"/pictures/nww_" + imgName + ".on.gif"; } function imgInact(imgName) { document[imgName].src =
"/pictures/nww_" + imgName + ".gif"; } //-- [FrontPage HTML Markup Component][FrontPage Component]
```

See the code of the JavaScript:

```
<script language='JavaScript'><!--  
function imgAct(imgName)  
{  
  document[imgName].src = "/pictures/nww_" + imgName + "on.gif";  
}  
function imgInact(imgName)  
{  
  document[imgName].src = "/pictures/nww_" + imgName + ".gif";  
}  
//--></script>  
<a href='#name' onMouseOver='imgAct("brow")' onMouseOut='imgInact("brow")'><img alt='Browse'  
name='brow' src='/pictures/nww_brow.gif' border='0' width='129' height='21'></A>
```

There are two parts:

- the **functions** changing the source of an image and
- the **call** of the different functions.

If scripting is not allowed, only the up picture can be seen. This is because every browser ignores unknown tags. The coding cannot be seen, because it is hidden inside a comment. The closing HTML comment has to start with an JavaScript comment!

If scripting is allowed, the script is loaded. It is necessary to name the img tag to access this image later on. If now the mouse is dragged over the image, the onMouseOver event occurs and the function imgAct("brow") is executed. This function now replaces the source of the image "brow" with the image "/pictures/nww_" + "brow" + "on.gif". If the mouse leaves the image, an onMouseOut event is sent and imgInact("brow") is executed. Now the picture "/pictures/nww_" + "brow" + "on.gif" will be loaded.

As the example shows, two pictures for the different state of the button are needed. The name of the image is generated from the img tag name and an individual extension.


One question remains: How does this work together with a form?

JavaScript offers the possibility of calling a method that behaves like a pressed submit button. An anchor can call a function instead of linking to another URL. Using this leads to the following script:

```
<script language='JavaScript'><!--
function goon()
{
    document.FMENU.submit();
}
function imgAct(imgName)
{
    document[imgName].src = "/pictures/nww_" + imgName + ".on.gif";
}
function imgInact(imgName)
{
    document[imgName].src = "/pictures/nww_" + imgName + ".gif";
}
//--></SCRIPT>
<form method='get' action='/cgi-bin/nwwcgi.exe/sysweb/nat-env ' name='FMENU'>
<a href='javascript:goon()' onMouseOver='imgAct("brow")' onMouseOut='imgInact("brow")'><img alt='Browse'
name='brow' src='/pictures/nww_brow.gif' border='0' width='129' height='21'></a>
</form>
```

Again the form tag has to be **named** to access later. The **href** attribute of the anchor now calls the function **goon()** which calls the **submit()** method of the form tag.

Now putting everything together, our layout for the main page looks like this:

Maintenance of File		
MENU		
Back ← Home... Select... Browse... Listing... New...		Data Source
		File: <input type="text" value="EMPLOYEES"/>
		Selection by Name
		From: <input type="text" value="A"/>
		To: <input type="text" value="D"/>
Natural		

Functional Parts



How to Use Frames

It is easy to set up frames, but frames cannot be bookmarked. It is possible to bookmark subpages of the frame, but you cannot start up the whole frame again.

Instead of frames, tables can be used for page formatting. The different parts of a page can be generated by separate subprograms. The resulting page is usually compatible with the frame-based page.

How to Use JavaScript

Using JavaScript on your page can be very useful. Use special script files and link these files to your application with the "SRC" attribute of the script tag.

Scripting on mainframes can cause specific problems. The character [] { } may not be defined in the EBCDIC character set used or cannot be typed on the terminal used.

If you use the Natural Web Server Extensions, a special feature can translate the &#..; equivalent back to the characters before sending to the HTTP server.

Do not forget to specify a <NOSCRIPT> section for older browsers.

If scripting is not allowed, you may not allow the user to go on or you may specify alternative pages.